T-61.5020 Statistical Natural Language: Course assignment

Comparing methods to classify texts according to the personality of its author

Géraud Le Falher—336 978, GERAUD.LEFALHER@AALTO.FI

May 31, 2013

Abstract

In this project, I studied how free texts can be analyzed to determine the five major traits of the author's personality. After carefully extracting a term document matrix using stemming, I compared the classification performance of three methods, Support Vector Machine, MultiNomial Naive Bayes and k-Nearest Neighbors before concluding that the task is difficult. Indeed, the best accuracy achieved overall is 62.9%.

1 Introduction

Human personality can be described in terms of five *traits*[1], presented in [4] as follow:

- Extroversion vs. Introversion (sociable, assertive, playful vs. aloof, reserved, shy)
- Emotional stability vs. Neuroticism (calm, unemotional vs. insecure, anxious)
- Agreeableness vs. Disagreeable (friendly, cooperative vs. antagonistic, faultfinding)
- Conscientiousness vs. Unconscientious (self-disciplined, organized vs. inefficient, careless)
- Openness to experience (intellectual, insightful vs. shallow, unimaginative)

In this project, we will consider that each of these dimensions is binary (whereas one may argue that no individual can be perfectly extroverted or introverted) and we will try to classify people in each dimension based on their writing. More specifically, some students were asked to produce a so called *stream of consciousness* essay, meaning that they wrote their current thoughts freely for twenty minutes. Pennebaker, King, *et al.* [6] have collected 2468 such essays which account for about 1.6 million words.[2] Furthermore, they have labeled this dataset by assessing the personality of each author with a standard questionnaire.

Psychology has showed that the way we express ourselves (for instance by writing) reflects our personality. A summary of these findings in the case of extroversion is given in[4, Table 1]. For instance, introverts tend to use a more diverse lexicon, more elaborated constructions but less positive emotion words. In Bayesian terms, we would say that the text is an observed variable which is conditioned by a hidden one, the personality. It is then quite natural to use a statistical natural language processing approach, first by extracting relevant features (described in Section 2 on the following page) and then training classifiers of a different kind like MultiNomial Naive Bayes (MNNB), k Nearest Neighbors (KNN) and Support Vector Machine (SVM) (as explained in Section 3 on page 5).

In itself, this problem is not of much interest, first because it is not a very convenient for the people being assessed to spend twenty minutes writing a text and also because this information alone is of little use. Yet it is interesting as a sub routine for a higher level task like matching users in a dating site or forming team of workers. It may also affect the choice of a more refined model for further mining of an individual's texts. Finally, it can be thought as a generalization of sentiment analysis, where instead of deciding between rather straightforward and factual classes (positive or negative), more complex facets of expression are analyzed.

[1] Although others are listed on the relevant Wikipedia page.
[2] It is surprisingly difficult to come with a precise figure because *word* is loosely defined.

2 Pre processing

Before doing any classification, I performed several operations to transform the raw data contained in the file `essays.csv` into a document-term matrix, which is a more suitable representation.

- First, in each line, I separated the text itself from the five labels, which posed no difficulties but is mentioned here for the sake of completeness. Another "easy but annoying" issue was that some characters were generating encoding errors (which was solved not elegantly by removing them, since there was only a few of them).
- I then wondered what to do about numbers not written in full. I changed single digit into equivalent word (as shown in Table 1) and replaced all the others by a single unique token (`xnumx`).

	zero	one	two	three	four	five	six	seven	eight	nine	*total*
raw	18	4,816	1,193	518	287	276	126	95	64	60	7,453
converted	134	5,090	1,814	1,069	737	748	375	297	309	262	10,835

Table 1: Counts of the ten words representing digits. The first line refers to the raw data, whereas in the second, single digit number have been converted to the corresponding word. Although it is probably irrelevant, it is amusing to note most people in this informal setting write numbers with digit and not letters, especially if the number is not 1 (and to some extent, 2 and 3).

- To reduce the sparsity of data, I decided to stem all the words, even though it was not such a severe problem because the texts were all in American English, which is a rather analytical language. Because I used the python language, I first looked at various algorithms offered by the Natural Language ToolKit (NLTK) library[1]. `nltk.WordNet` is based on the morphy function[3] that applies some suffix-suppression rules before looking up in a database of base forms. In my case, it was rather slow and for some reasons, it only removed plural endings (like *s*) but did nothing about past tense verbs. NLTK also implements Porter[8] and Snowball[7] algorithms but although it is highly subjective, I found them somewhat too "aggressive", for instance transforming *was* to *wa*. Therefore I finally chose hunspell[4], which nonetheless came with its own issues such as totally discarding punctuation, segmenting differently (for instance, *mid-day* to *mid* and *day*) and generating some irrelevant alternatives (*thing* is transformed into *thing* but also into *the+ING*) or missing one (*woke* was not changed to *wake*). For the last two problems, Part Of Speech (POS) tagging could have helped but because of the

[3] http://wordnet.princeton.edu/man/morphy.7WN.html
[4] http://hunspell.sourceforge.net/

other issues, the two version of the text were no more aligned. Still in regard with sparsity, it reduces the number of unique tokens from 29,535 to 13,407.

- I also collected some general characteristics of the text, namely: the number of sentences, words and characters; the proportion of punctuation marks and capitalized letters; and the number of words per sentences.

- The use of POS can be indication of the personality. For instance, introverts use more nouns while extroverts favor verbs. Thus I tagged every word with the default tagger of NLTK (based on a maximum entropy model) (I also considered using a Conditonal Random Fields (CRF) implementation [for instance 5] but it required too much training for my goal). It turned out it was the most time consuming operation (more than 23 minutes on my laptop). After that, I counted in every text how many times each of the 27 tags appears. A sample is shown in Table 2.

	PRO	N	V	P	ADV	.	DET	ADJ
total	263,748	260,182	255,939	169,896	154,300	121,431	114,085	90,769
text 1515	189	85	112	135	78	149	194	87
text 1789	108	131	57	86	130	31	69	81
text 1804	38	21	68	35	58	57	24	44

Table 2: Repartition of the eight most represented tag in total and for three random texts.

- One thing that I have not had time to consider is Named Entity Recognition. Yet it would have been interesting to see if this kind of information provide some insight about personality. For instance, we may imagine that someone citing a lot of different places is more likely to be classified as opened to experience.

- The last step was to go through all the texts to find all the distinct tokens along their counts. A sample of that is shown in Table 3 on the following page. Using this list, I processed every text individually to compute its feature vector, that consists of:
  - the five class label
  - the six general characteristics
  - the count of each of the 27 part of speech
  - the count of each of the 13,407 token

i	to	the	and	that	my	a	it	is	of	t
122,593	56,646	40,466	38,077	31,740	29,830	29,153	27,560	25,299	23,177	20,466

Table 3: The first 11 of the 13,407 unique tokens. As expected, most of them are stop words, expect for *I*, which is explained by the nature of a stream of consciousness essays and *t* which comes from negation's contraction, as the texts are written rather informally.

3 Classification

Another way to alleviate the sparsity problem is to resort to general methods of dimensionality reduction such as Self-Organizing Map (SOM) or Independent Component Analysis (ICA). Due to lack of time, I only experimented with Latent Semantic Indexing (LSI), which consist of computing a Singular Value Decomposition (SVD) of the document matrix W and keeping only the r largest values.

MultiNomial Naive Bayes A simple model which assume that all features are mutually independent and only depend of the class, i.e. $P(c \mid w) \propto P(c) \prod_i P(w_i \mid c)$. Because I use a bag-of-word model, $P(w_i \mid c)$ are supposed to follow a multinomial distribution whose parameters are simply the count in the data, according to the maximum likelihood estimation. I use the off-the-shelf implementation of MATLAB[5], whose interface is very simple.

Support Vector Machine A maximum margin separator that operates in a high dimensional space derived from the original one by the use of a kernel. There, the two classes are linearly separated by a hyperplane described by the so called "support vector" instances, selected by the optimization procedure. It works because kernel is a function $K(x, y)$ which measure the similarity between two instances x and y: the smaller it is, the more related they are. The easiest choice was to use a kernel based on euclidean distance (namely, a radial basis function $K(x, y) = e^{-\frac{||x-y||^2}{\sigma^2}}$) but it would have been more interesting to implement a common subsequence kernel[3], $K_n(x, y) = \sum_{s_x \in \mathcal{S}_n(x)} \sum_{s_y \in \mathcal{S}_n(y)} \mathbb{1}_{s_x = s_y}$, that, as its name suggest, keep track of how many subsequences of length n are shared by x and y. On a practical note, I used a C++ implementation written by Chang and Lin [2] with a MATLAB interface. One flaw of my approach was that although SVM requires careful tuning of its two parameters σ^2 and C (for regularization), it is very time consuming so I did it once for one dimension and then use the same parameters for all subsequent runs.

[5] http://www.mathworks.se/help/stats/naivebayes.fit.html

k Nearest Neighbors Contrary to the two other methods, kNN has no training phase but when given a new sample to classify, it finds the k closest points in the training set and assign their majority class. The hyperparameters of the model are k (which I set to 5) and again, the distance used. Using the MATLAB implementation[6], I had the choice between ten of them,[7] and after some preliminary comparison, I chose correlation distance, which is one minus the linear correlation coefficient between two documents seen as a sequence of numerical value. Because I found out it did not hurt accuracy yet it was beneficial to the speed, I used only tokens that appeared in more than 3% and less than 90% of all documents, which gave 1,052 features.

Since the dataset was almost perfectly balanced for every five traits, I decided that accuracy was a sufficient measure of assessing performances of the various methods and distinguishing them from random guessing. The other noticeable methodology point is that all measures reported have been computed from 5-fold cross validation.[8]

Because SVM performs better when all the features are in the same range, I first planned to use tf-idf weighting with a `llc` scheme. Namely, if $N = 2468$ is the number of documents, $a_{i,j}$ the number of times that word j occurs in document j and $d_j = \sum_{i=1}^{N} \mathbb{1}_{a_{i,j}>0}$ the number of documents that contain the word j appear, I replaced $a_{i,j}$ by $b_{i,j} = (1 + \log(a_{i,j})) \log \frac{N}{d_j}$ and then I normalized the column of B to unit vector (or set it to zero if the word appears in all document[9]). But after looking at the initial results, I simply stuck with *standardization*, that is subtracting the mean and dividing by the standard deviation of each column.

	Extroversion	Neuroticism	Agreeableness	Conscientiousness	Openness
	Full matrix				
MNNB	0.564	0.587	0.563	0.552	0.629
SVM	0.557	0.532	0.524	0.528	0.597
kNN	0.533	0.521	0.481	0.514	0.557
	250 most relevant dimensions				
MNNB	—	—	—	—	—
SVM	0.550	0.551	0.542	0.517	0.573
kNN	0.518	0.522	0.509	0.509	0.526

Table 4: Accuracy of the three methods for the five traits, using or not dimensionality reduction (the multinomial assumption does not hold after the SVD reconstruction).

[6] http://www.mathworks.se/help/stats/classificationknn.fit.html
[7] I discard the Mahalanobis distance because computing the covariance matrix took too much time.
[8] For even more details, the code is available on github: https://github.com/daureg/wcpr13
[9] Although it did not happen in this dataset.

The first comment that the results presented in Table 4 on the previous page calls is that this kind of classification is "difficult" in the sense that in most cases, the accuracy is only slightly above what random guessing would have generated. The second is that dimensionality reduction does not produce a very noticeable impact on the results. Then, *Openness* is significantly easier to predict than the other traits for all the methods. Finally, despite its conceptual simplicity, MNNB consistently outperforms SVM while kNN is the worst of the three, probably because the metric used is not well adapted to text.

4 Conclusion

Predicting author's personality from one of their stream of consciousness essays is a difficult task that is naturally suited to Statistical Natural Language Processing (SNLP). After some pre-processing steps (text extraction and cleaning, stemming, POS tagging and term-document matrix building), I tried three machine learning methods to perform classification. Naïve Bayes was the most successful although the others may have benefit from the use of a distance metric tailored to text analysis.

References

[1] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. O'Reilly Media, 2009.

[2] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1–27:27, 3 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[3] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels", *Journal of Machine Learning Research*, vol. 2, pp. 419–444, Mar. 2002, ISSN: 1532-4435. [Online]. Available: http://jmlr.org/papers/v2/lodhi02a.html.

[4] F. Mairesse, M. A. Walker, M. R. Mehl, and R. K. Moore, "Using linguistic cues for the automatic recognition of personality in conversation and text", *Journal of Artificial Intelligence Research*, vol. 30, no. 1, pp. 457–500, 2007.

[5] N. Okazaki, *Crfsuite: a fast implementation of conditional random fields (crfs)*, http://www.chokkan.org/software/crfsuite/ [Last accessed: 2013-05-27], 2007.

[6] J. W. Pennebaker, L. A. King, *et al.*, "Linguistic styles: language use as an individual difference", *Journal of personality and social psychology*, vol. 77, no. 6, pp. 1296–1312, 1999. DOI: 10.1037/0022-3514.77.6.1296.

[7] M. Porter, *Snowball: a language for stemming algorithms*, http://snowball.tartarus.org/ [Last accessed: 2013-05-27], 2001.

[8] M. F. Porter, "An algorithm for suffix stripping", *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, 1980. DOI: 10.1108/eb046814.

List of Acronyms

CRF	Conditonal Random Fields. 4
ICA	Independent Component Analysis. 5
κNN	k Nearest Neighbors. 2, 6, 7
LSI	Latent Semantic Indexing. 5
MNNB	MultiNomial Naive Bayes. 2, 6, 7
NLTK	Natural Language ToolKit. 3, 4
POS	Part Of Speech. 3, 4, 7
SNLP	Statistical Natural Language Processing. 7
SOM	Self-Organizing Map. 5
SVD	Singular Value Decomposition. 5
SVM	Support Vector Machine. 2, 5–7